Remarks

Entry of the amendments, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. After entry of the amendments, claims 1, 3, 5 & 7-24 are pending.

By the above amendments, independent claims 1, 3 & 5 are amended to more clearly point out and distinctly claim certain aspects of applicants' invention. Applicants note that although "one or more resources" is replaced with "a resource," it is for purposes of clarity, and the claimed technique of managing the locking of resources remains applicable to one or more resources. These amendments also incorporate, in part, subject matter from dependent claims 2, 4 & 6. Additional support for the above amendments can be found throughout the application (see, e.g., p. 10, line 26 – p. 11, line 18; p. 12, lines 4-18; p. 13, line 11 – p. 15, line 20 of applicants' specification). Further, since dependent claims 2, 4 & 6 are canceled herein, references to these claims in claims 7, 13 & 19, respectively, are amended appropriately. Thus, no new matter is being added to the application by the amendments presented herewith.

In the Office Action dated August 6, 2003, claims 1-24 are rejected under 35 U.S.C. §102(e) as being anticipated by Traversat et al. (U.S. Patent No. 6,119,129; hereinafter, "Traversat"). Applicants respectfully, but most strenuously, traverse this rejection to any extent deemed applicable to the claims presented herewith.

In one aspect, applicants' invention is directed to multithreaded clients accessing resources of a global data repository. For example, applicants claim a method of managing locking of resources of a global data repository of a distributed computing environment (e.g., claim 1). The method includes, for instance, issuing a request, via a thread of a multithreaded client application of the distributed computing environment, for a lock of a resource of the global data repository; and obtaining the lock for the thread independent of a threading model of an operating system of the distributed computing environment. The obtaining includes employing a local tree in obtaining the lock, the local tree being local to the client application and having a mount point usable by the client application to lock the resource. The resource is further lockable via another mount point of the local tree or another local tree.

Advantageously, employing local trees allows applications to access a resource (e.g., a table) via a mount point (e.g., directory) of a local tree even though the resource is locked in a global data repository (e.g., global database). Further, the same resource (table) can be mounted and locked through different mount points (directories). For instance, with a global database including Table X, and a local tree including directories A and B, an application can mount and lock Table X through Directory A, and mount and lock Table X through Directory B. Therefore, in this example, Table X is mounted and locked more than once, as if it were two different resources.

The use of a local tree is recited in applicants' claimed invention. For example, applicants' claim 1 includes a local tree that is local to a client application and has a mount point usable by the client application to lock a resource, the resource also being lockable via another mount point of either that local tree or another local tree. Applicants respectfully submit that at least these aspects are not taught or suggested by Traversat.

Traversat describes a journaling mechanism that maintains a detailed journal of all modifications made to a global configuration database (see Abstract thereof). In the context of this journaling mechanism, Traversat discloses locks relative to a global tree structure in a global configuration database (see, e.g., col. 8, lines 21-24 and lines 63-67). A lock in Traversat is globally applied to a resource so that database transactions are provided with one view of what is globally accessible in the database (see, e.g., col. 8, lines 50-67). This locking in Traversat is quite different from the obtaining of locks recited by the claims presented herewith.

For example, applicants recite employing a local tree (i.e., local to a client application) in obtaining a lock, wherein the local tree has a mount point usable to lock a resource which is further lockable via another mount point of either the local tree or another local tree. This locking from different mount points allows, for example, the same table to be locked from different directories. The mounting mechanism of the claimed invention also provides an application with a flexible, rather than a uniform, view of resources of the global data repository. For example, an application's multiple libraries can each lock the same table via different directories, thereby allowing the same table to serve the unique purposes of each library.

In contrast, Traversat lacks a description or suggestion of a local tree having a mount point usable by a client application to lock a resource, wherein the resource is also lockable via another mount point of either that or another local tree, as recited by the present invention. Since the Traversat patent globally applies a lock relative to the database, it has no need for a resource being locked via a locally-based mount point, such as a mount point of a local tree, let alone a resource being locked via different mount points.

Further, Traversat does not include any teaching or suggestion of a local tree that is local to a client application. Traversat explicitly avoids the use of trees local to client applications at col. 4, lines 10-16:

> In the described embodiment, data relating to client machine and user configuration in a network is stored in on [sic] a server as part of a server JSD. The configuration information for each client, also referred to as subsystem, is stored in the server schema. This is in contrast to conventional networks where configuration information regarding a client is hardcoded or stored on the client machine.

Still further, Traversat applies locks to provide transactions with a uniform view of the configuration database, rather than the flexible view provided by the capability of locking a resource using a mount point and another mount point, as recited by the claims presented herewith. For example, the commit and abort transactions in Traversat uniformly release locks and all waiting threads are provided with a uniform view of what is accessible in the database (col. 8, lines 21-24).

The Office Action states that Traversat teaches the use of a local tree in obtaining a lock and cites col. 8, lines 21-24. To the extent that this cited section is applicable to the claims presented herewith, applicants respectfully submit that this section does not teach or suggest a local tree that is local to a client application. The cited section describes transactions by which "locks on root nodes or leaf nodes in the database are released" (col. 8, lines 22-23). It is clear from statements immediately preceding and following the cited section that "the database" refers to the server-based configuration database (see, e.g., col. 8, lines 18-21 and col. 8, lines 28-29). The tree structure being locked in the configuration database in Traversat is a global tree rather

than a local tree, and resides on a server which is separate from client machines and client applications (see col. 8, lines 28-29 and FIG. 1 thereof). Again, as described above, Traversat explicitly avoids the use of trees local to client applications. Thus, the tree in Traversat is not local to a client application, as recited by the claims presented herewith.

Not only does Traversat fail to teach or describe employing the local tree local to the client application and the mounting and locking techniques as described above, it also fails to even suggest these features. Traversat's locks are directed to a journal mechanism that logs database transactions, including those that process locks (see Abstract and col. 8, lines 50-67). The journal mechanism of Traversat resides outside the global configuration database (col. 7, lines 23-30). Being separate from the database, the journal mechanism requires additional management and maintenance, and is an additional point of failure in the system (e.g., the journal mechanism may fail or the part of the system that writes to the journal may become corrupted).

In contrast, the present invention addresses a distributed computing environment, which includes, for example, a highly available cluster environment. Such a highly available environment requires minimizing points of failure. Applicants' recited obtaining of locks employing local trees that are mounted multiple times provide a locking mechanism without adding an additional point of failure, such as a journal mechanism. Thus, applicants' recited obtaining of a lock and the local tree being mounted multiple times provide a simple technique that enhances the management of resource locking in a distributed computer environment.

Thus, applicants respectfully submit that the locking technique in Traversat, being based on a journal system that does not minimize points of failure, is not even suggestive of the above-described recited features of the present invention.

For all the reasons stated above, applicants respectfully submit that claim 1, as well as claims 3 and 5, include at least one feature not described, taught or suggested by Traversat. For example, Traversat fails to describe, teach or suggest employing a local tree to obtain a lock of a resource of a global data repository, wherein the local tree is local to the client application and wherein the local tree has a mount point usable by the client application to lock the resource, wherein the resource is further lockable via another mount point. Thus, applicants respectfully

request an indication of allowability for claims 1, 3 & 5. The dependent claims are allowable for the same reasons as the independent claims, as well as for their own additional features.

Based on the foregoing, applicants respectfully request an indication of allowability for all pending claims.

Should the Examiner wish to discuss this case with applicants' attorney, please contact applicants' attorney at the below-listed number.

Respectfully submitted,

Blanche E. Schiller

Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: December 8, 2003.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579